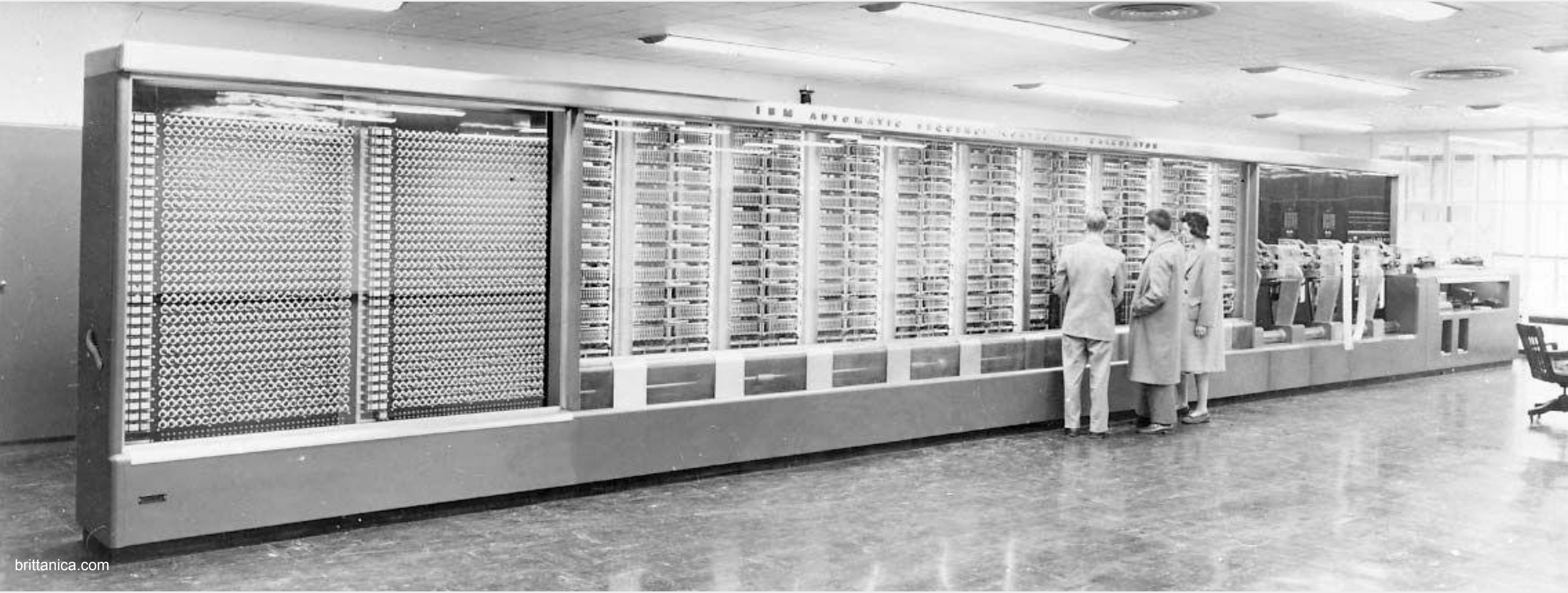
Compiling Techniques

Lecture 1: Introduction

Wikipedia, Public Domain

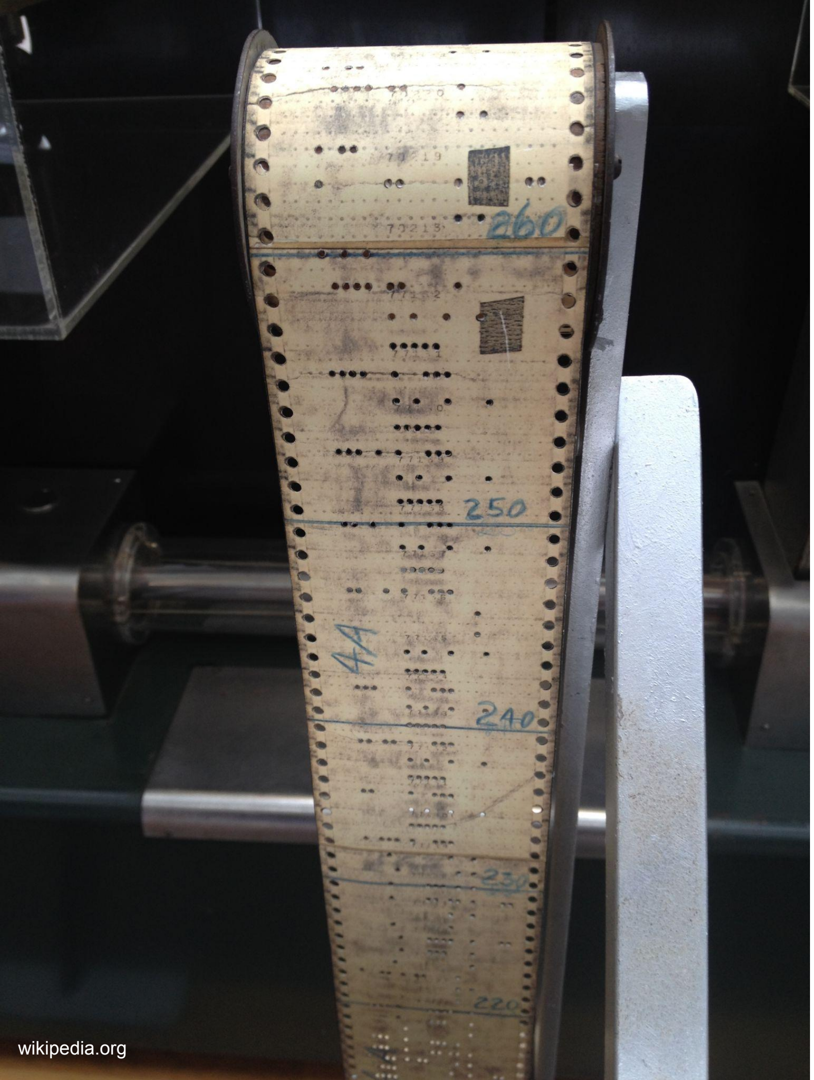

GRACE
HOPPER
IS THE
QUEEN OF
COMPUTERS
LETTERMAN

Harvard Mark I, 1943



brittanica.com

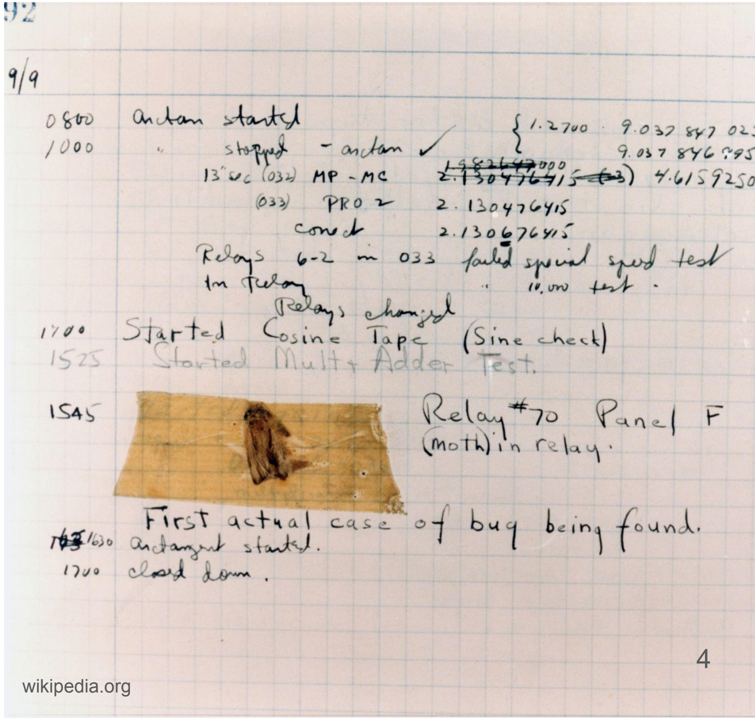
Program Tape of The Mark I



wikipedia.org

A Patch!

First Bug!



wikipedia.org

Grace Murray Hopper

- 1906 Born — 1905, first practical plane
- 1934 PhD in Mathematics @ Yale
  (very unusual for women at this time)
- 1943 US Navy to work on Mark I
- 1946 Wrote **Mark I Manual**
  https://chsi.harvard.edu/harvard-ibm-mark-1-manual
- 1952 Developed the **first compiler** (for the A-0 system)
- 1959 Defined **COBOL** (as part of a larger committee)
- … ⇨ https://www.youtube.com/watch?v=wEC30qhXPp0

Grace Murray Hopper Collection, 1944-1965, Archives Center, National Museum of American History

Lecturer

Amir Shaikhha

Lecturer in Data-centric Systems

https://amirsh.github.io

Essentials

Website — [Learn: Compiling Techniques (2023-2024) [Sem 2]]

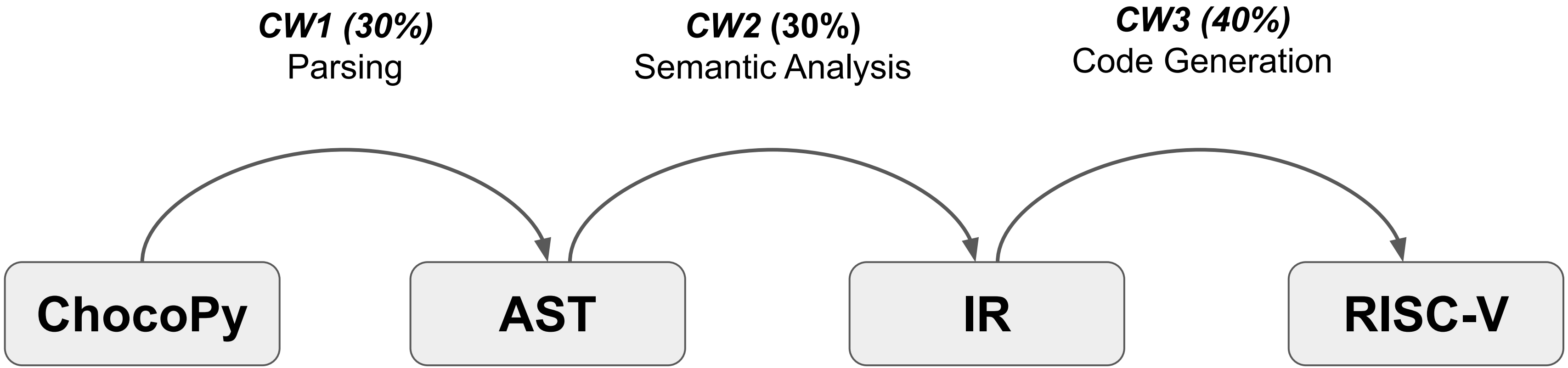
Discussions — Follow link “Discussions (Piazza)” on Learn

Textbook — Keith Cooper & Linda Torczon: Engineering a Compiler Elsevier (not strictly required)

Essentials

- Course is **20 credits**
- Evaluation
  - No exam ⇒ **Coursework only**
- A lot of programming
  - A lot of hours on coursework
  - Python is the primary language we use
- Each week
  - 3h lectures
    - Monday 15:10 - 16:30, Appleton Tower, Lecture Theater 3
    - Thursday 15:10 - 16:00, Appleton Tower, Lecture Theater 3
  - 2h labs, Thursday 16:10 - 17:30, Appleton Tower, 6.06

Coursework: A Python to RISC-V Compiler



Coursework Schedule

Week 1 (Jan 15)		Week 6 (Feb 26)	***CW2***
Week 2 (Jan 22)		Week 7 (Mar 4)	
Week 3 (Jan 29)		Week 8 (Mar 11)	
Week 4 (Feb 5)	***CW1***	Week 9 (Mar 18)	***CW3***
Week 5 (Feb 12)		Week 10 (Mar 25)	
Learning Week		Week 11 (Apr 1)	

Deadlines: Friday noon

Marking and Autograding

Labs

- Will help you with coursework

- 1 session of 2 hours

- Start: Week 3
- End: Week 11

- Time: Thursday 16:10 - 17:30
- Location Appleton Tower, 6.06

Coursework is Rewarding

You will understand what happens when you type:

```
$ python program.py
```

But also:

- Will deepened your understanding of computing systems (from language to hardware)
- Will improve your programming skills
- Will learn about using revision control system (git)

Class-taking Technique

- Extensive use of projected material
  - Attendance and interaction encouraged
  - Feedback also welcome
- Reading book is optional
  (course is self-contain, book is more theoretical)
- Not a programming course!
- Start the practical early
- Help should be sought on Piazza in the first instance

Syllabus

- Overview
- Scanning
- Parsing
- Abstract Syntax Tree
- Semantic analysis
- Code generation
- Real machines assembly
- Advanced topics
  - Instruction selection
  - Register allocation

Compilers

What is a Compiler?

A program that *translates* an executable program in one language into an executable program in another language. The compiler might improve the program, in some way.

What is an Interpreter?

A program that directly *executes* an executable program, producing the results of executing that program

Examples:

- C and C++ are typically compiled
- R is typically interpreted
- Java and Python are compiled to a bytecode and then either interpreted or compiled.

A Broader View

Compiler technology = Off-line processing

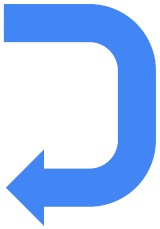
- Goals: improved performance and language usability
- Making it practical to use the full power of the language
- Trade-off: preprocessing time versus execution time (or space)
- Rule: performance of both compiler and application must be acceptable to the end user

Examples:

- Macro expansion / Preprocessing
- Database query optimisation
- Javascript just-in-time compilation
- Emulation: e.g. Apple's Intel transition from PowerPC (2006)

System Stack

Problem
Algorithm
Program (Language)
Runtime System (OS)
ISA (Architecture)
Micro-Architecture
Logic
Circuits
Electrons



Compilation

Why Study Compilation?

- Compilers are important system software components:
  they are intimately interconnected with architecture, systems, programming methodology, and language design
- Compilers include many applications of theory to practice:
  scanning, parsing, static analysis, instruction selection
- Many practical applications have embedded languages:
  commands, macros, formatting tags
- Many applications have input formats that look like languages:
  Matlab, Mathematica
- Writing a compiler exposes practical algorithmic & engineering issues:
  approximating hard problems; efficiency & scalability

Intrinsic Interest

Ideas from many different areas of computer science!

Artificial Intelligence	Greedy algorithms Heuristic search techniques
Algorithms	Graph algorithms Dynamic programming
Theory	DFA & PDA, pattern matching, Fixed-point algorithms
Systems	Allocation & naming, Synchronization, locality
Architecture	Pipeline & memory hierarchy management Instruction set
Software Engineering	Design pattern (visitor) Code organisation

Intrinsic Merit

Compiler construction poses challenging and interesting problems:

- Compilers must do a lot but also run fast
- Compilers have primary responsibility for run-time performance
- Compilers are responsible for making it acceptable to use the full power of the programming language
- Computer architects perpetually create new challenges for the compiler by building more complex machines
- Compilers must hide that complexity from the programmer
- Success requires mastery of complex interactions

Making Languages Available

It was our belief that if FORTRAN, during its first months, were to translate any reasonable ”scientific” source program into an object program only half as fast as its hand coded counterpart, then acceptance of our system would be in serious danger. . . . I believe that had we failed to produce efficient programs, the widespread use of languages like FORTRAN would have been seriously delayed.
John Backus (1978)

Next Lecture

The View from 35000 Feet

- How a compiler works
- What I think is important
- What is hard and what is easy